Information Flow Audit for Transparency and Compliance in the Handling of Personal Data

Information Flow Audit for Transparency and Compliance in the Handling of Personal Data

Thomas F. J.-M. Pasquier
Computer Laboratory, University of Cambridge
Cambridge, United Kingdom
Email: tfjmp2@cam.ac.uk

David Eyers
Department of Computer Science, University of Otago
Dunedin, New Zealand
Email: dme@cs.otago.ac.nz

***Abstract*—The adoption of cloud computing is increasing and its use is becoming widespread in many sectors. As the proportion of services provided using cloud computing increases, legal and regulatory issues are becoming more significant. In this paper we explore how an Information Flow Audit (IFA) mechanism, that provides key data regarding provenance, can be used to verify compliance with regulatory and contractual duty, and survey potential extensions. We explore the use of IFA for such a purpose through a smart electricity metering use case derived from a French Data Protection Agency recommendation.**

I. Introduction

In a recent paper Singh et al. [1] explored the use of Information Flow Control (IFC) as a data-centric means to control data flows within the cloud in order to meet *"civil, administrative, and criminal law obligations and responsibilities"*. The enforcement of IFC is mandatory on every data exchange between system components within and across machines. Through being data-centric, IFC enforcement is easier to implement consistently than *ad hoc* access control checks within software. Among other things, IFC can guarantee that data disclosure constraints are respected [2], for example, that medical data can only be used in authorised medical contexts. It may also guarantee that proper procedures are followed when transferring data across context domains, e.g. medical data must receive consent for release and be anonymised before it can be used in a research context.

Pasquier et al. [3] described a mechanism in which IFC can be applied in a Platform as a Service (PaaS)[1] cloud context. The IFC enforcement mechanism is the responsibility of the cloud provider, underlying yet being totally separated from application logic. A properly implemented IFC mechanism allows applications running on top of the enforcement mechanism to collaborate without requiring mutual trust [4]. The only required trust relationship that must exist is between the enforcement mechanism provider and individual applications.

However, while IFC enforcement is a step towards demonstrating compliance with regulation requirements, a mechanism to provide transparency over data usage and processing is needed to complement IFC. Indeed, Bier [5] argues for the integration of usage control and provenance. Information Flow Control and Audit has been introduced by Pasquier et al. [6] to complement IFC to facilitate insight into data provenance [7] through visualisation and analysis of how information has actually flowed across the system.

In this paper we explore how such a mechanism can be developed as a tool to help in the understanding of issues such as potential data leakage, or attributing responsibilities. The paper aims to be intelligible by both computer scientists and law scholars. In §II, we briefly discuss the legal context. In §III, we briefly explain the concept of Information Flow Control & Audit as well as pointing to the relevant technical literature. In §IV, we explore the application of Information Flow Audit to a electricity smart metering scenario. In §V, we present related work, discuss open challenges and survey a range of potential solutions in existing academic work. Finally we summarise our conclusions in §VI.

II. Motivation

It is important to understand the roles of the actor involved in the handling of personal data. The UK Information Commissioner defines [8]: 1) the *data controller* as *"a person who (either alone or jointly or in common with other persons) determines the purposes for which and the manner in which any personal data are, or are to be processed"*; 2) the *data processor* as *"in relation to personal data, means any person (other than an employee of the data controller) who processes the data on behalf of the data controller"*; and 3) *processing* as *"obtaining, recording or holding the information or data or carrying out any operation or set of operations on the information or data"*.

Data protection laws require the data controller to comply with principles such as justifiable data processing or implementation of proper security measures. Further, some data may be considered particularly sensitive, for example medical data that requires explicit consent for it to be released for processing.[2] The data controller must retain the responsibility for the proper handling of data. The processor must only process the data in the context specified by the controller and must not retain the data beyond the time necessary for the purpose of processing. Further, the data processor should contractually be required to implement proper technical or organisational security/privacy measures.

However, the lack of transparency of most cloud processing offers make assessing the proper handling of data difficult.

[1] PaaS is a category of cloud service allowing tenants to develop, run and manage cloud applications. The cloud service provider manages the complexity of building and maintaining the underlying infrastructure which is abstracted from the tenants.

[2] Article 29 Data Protection Working Party 2015 – European Commission

Traditional audit trails are provided by legacy applications and tend to focus exclusively on the internal working of said application [9]. In such a scenario it becomes difficult to understand how an individual data item has been handled and if it complies with regulation. Moreover, relationships involving subcontracts and other forms of interaction across multiple parties complicate the problem further. Audit must be data-centric and cross-application, providing information on the flow of a particular data item during its whole life cycle, allowing assessment of where responsibility for that data item lies. This audit trail represents the actions performed on data and the entities that are responsible for those actions [10].

III. Information Flow Control & Audit

The early days of the Internet with its prevalent, simple, client-server architecture are long gone. Most applications are composed of multiple services interacting through Application Programming Interfaces (APIs). Furthermore, applications may exchange information to deliver services to the end-user. The different services and applications may fall under different management regimes and legislatures. However, much of the underlying complexity of these software systems is not made explicit, and is thus poorly understood by most end-users. This can lead to significant issues in certain cases. For example, the personal data of EU citizens is required to be stored on infrastructure within the EU or in certain specified Safe Harbours,[3] therefore a complex chain—often undisclosed, or with details only implicitly specified—of third-parties may expose a company to legal pursuit if it has failed to implement proper handling of such personal data. What is needed is a mechanism for control and auditing that applies across application and service boundaries.

A. Information Flow Control

Information Flow Control (IFC) is such a mechanism. IFC is a data-centric, mandatory access control mechanism that guarantees non-interference across security contexts. IFC is continuously enforced during interactions at every information exchange. When enforced at the Operating System (OS) level, every system call is checked for IFC conformance, e.g. reading or writing a file.

It has been demonstrated that IFC helps in reducing the size of the Trusted Computing Base (TCB) [4]. When IFC is in place, there is no need to trust applications running above the level of IFC enforcement for proper data usage to be guaranteed [2]. For example, in the case of enforcement at the OS level, applications running on the OS need not be trusted [4]. A tamperproof mechanism enforcing IFC policy (be it the compiler, the kernel, etc.) controls every data exchange between entities, guaranteeing that data do not leave their designated security contexts.[4] The privilege to transfer data across security contexts is limited to a well defined number of trusted entities called *declassifiers* and *endorsers*, as illustrated in Fig. 1. The TCB is limited to the enforcement mechanism and to a lesser extent, to the declassifier/endorser, to transfer data to and from well-defined security contexts.

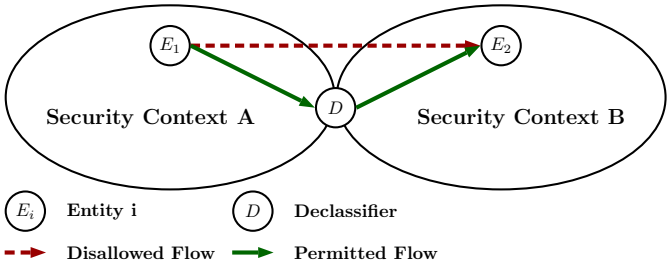

Fig. 1. Transferring data across security contexts through a Declassifier (or Endorser, not shown).

A large number of regulations relating to data management can be expressed as constraints over the flow of information within a system. Such constraints can be enforced by augmenting access control, encryption and containment with IFC. However, we believe that this is not yet sufficient to meet the needs of regulated sectors. There is a need for transparency when demonstrating compliance, which can be achieved by capturing and recording information flows. The record of such flows may allow us to demonstrate consistent application of regulations and policies, and to understand the chains of events that might lead to a system releasing data in a manner that is contrary to regulation.

Current application-centric logging mechanisms fail to provide such transparency. Indeed, those data logs tend to be composed of legacy and/or service-specific logging systems. Such logs typically focus on a specific software component of the system—for example, relating to a web server or a database—rather than providing an overview of the system behaviour with respect to its intended uses. They are difficult to interpret system-wide, as they tend to select for logging only those events relevant to the particular system component, and the log record is in component-specific terms. Aggregation mechanisms have been provided to process logs from several applications and several layers of the software stack. However, none can give satisfactory results in a cloud computing context, as the logged data are heterogeneous and often fail to capture information on specific items of data [9]. Given that we argued that enforcement of integrity and confidentiality issues should take a data-centric approach, a data-centric approach to audit and logging seems appropriate for the demonstration of compliance with data regulations. We believe that such a mechanism could emerge from provenance research.

B. Provenance Systems

Data provenance (sometimes called *lineage*), can be understood as a means to describe *where, when, how* and *by whom* data was generated or manipulated. Provenance data are generally used for verification of data quality, generation of replication recipes, attribution of ownership, understanding of context, determination of resource usage, and analysis of error in data generation [12]. Some of these uses of provenance can be directly associated with legal requirements.

[3]The notion of Safe Harbour itself is currently under review by the European Commission, following Maximillian Schrems v Data Protection Commissioner 23/09/2015 http://curia.europa.eu/juris/document/document.jsf?text=&docid=157862&pageIndex=0&doclang=EN&mode=req&dir=&occ=first&part=1&cid=191188.

[4]Trust in the enforcement can be achieved through review of the enforcement mechanism and remote attestation—this is discussed in [11].

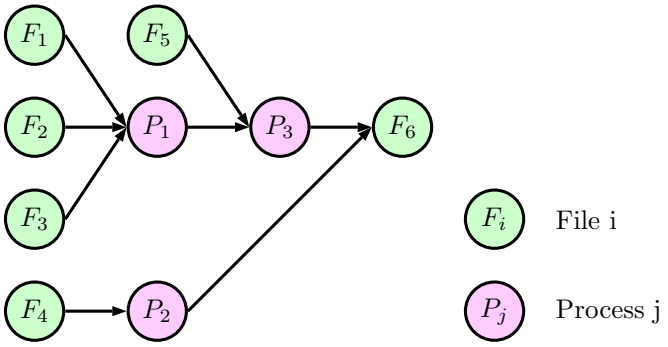

Fig. 2. Representing the relationship between information flows and entities (each File i and Process j) as a directed graph.

Provenance metadata is intended to represent how the data contained in a given entity was generated and how it relates to other entities in the system. This metadata is generally represented and analysed as a directed graph, as illustrated in Fig. 2.

System provenance was traditionally a focus of databases or data storage systems. For example, Muniswamy-Reddy et al. [13] record interactions between files and processes in the Linux OS. However, this fails to capture a certain number of important interactions in a complex real-life system. In order to understand how information is manipulated or generated, it is not sufficient only to monitor the interaction between a process and persistent data, but also requires the additional collection of data that captures the interactions of processes, inter-process communication mechanisms and the OS kernel via system calls. This has led to the emergence of whole-system provenance [14].

C. From Provenance to Information Flow Audit

Whole-system provenance [14], [15] and IFC OS-level enforcement often use the same mechanism to track exchanges of information between processes and other kernel objects. Furthermore, key concepts, vocabulary and understanding of the system are extremely similar.

During the enforcement of IFC, the data logged about the system behaviour allows a provenance graph to be built. Furthermore, as the enforcement mechanism and the audit mechanism are tightly coupled, the data captured can be tailored to focus on and meet the requirements of the policy being enforced. One of the main problems with provenance is the large storage overhead introduced by provenance data [16]. By focusing the information logged to that relevant to, and selected for some purpose, we can significantly reduce this storage overhead.

Information Flow Audit allows us to verify compliance by performing queries over the audit graph. A simple example would be a query verifying that there is no path from a medical database to a research database that does not pass through an entity that anonymises the data. In the case of data leakage, studying the paths from the data source may allow users to understand how data leaked, and to attribute responsibilities accordingly. We believe that Information Flow Audit can help in demonstrating compliance, assigning responsibility, helping investigation and generally improving transparency.

Further, in addition to active entities (e.g. a computer process) and passive entities (e.g. files, sockets, pipes etc.), audit records may also represent *agents* (i.e. a contextual entity acting as an enabler, catalyst, controller of a process execution) and *artefacts* (i.e. immutable digital or physical objects) as defined by the Open Provenance Model [17]. This allows the audit graph to represent interactions between natural[5] or legal[6] persons (i.e. agents), virtual entities and physical objects (i.e. artefacts). These relationships may help determine, for example, the ownership of a physical object or items of data, as well as who is controlling a particular computer process.

D. Cost of adoption

Any enforcement mechanism comes with an associated cost. Is it realistic to expect the adoption of techniques similar to those described in this paper? In terms of the performance of whole-system provenance, enforced at the operating system level, application performance overheads as low as 2.5% [14] and 2.7% [15] have been reported. Our implementation of IFC&A introduces a performance overhead of 3.6% [6]. Further, as discussed in [3], our proposed IFC&A implementation within the Linux kernel is transparent to most applications and supports legacy software without any reengineering effort. However, there is a cost incurred for the data storage that records captured audit data which, depending on the amount of sensitive data flow that needs to be tracked, may prove significant. This particular issue is further discussed in §V-D.

IV. Use-case

We explore the use of Information Flow Audit for demonstrating compliance by looking at a use case derived from a CNIL[7] report. The report[8] describes best practice for smart metering services for electricity supply, including those mediated by the cloud. We focus on the $IN \rightarrow OUT$ and $IN \rightarrow OUT \rightarrow IN$ scenarios. In the first scenario data are collected and processed in the cloud to provide services to the customer. In the second scenario, in addition, actuation commands may be sent to devices situated in the customer's house in order to control energy consumption. These scenarios and best practices would apply to products such as Nest thermostat devices.[9]

We extract four recommendations from the report:

1) anonymous data can be freely transferred to a third party;
2) personal data can be transmitted with explicit consent;
3) when a contract is terminated, data must be deleted, anonymised or archived (archiving is for litigation purposes and limited to a duration specified by law,

[5] i.e. a "real" human being.

[6] i.e. a private or public organisation.

[7] French National Commission on Information and Liberty.

[8] Pack de conformité sur les compteurs communicants, published in May 2014, available at http://www.cnil.fr/fileadmin/documents/Vos_responsabilites/Packs/Compteurs/Pack_de_Conformite_COMPTEURS_COMMUNICANTS.pdf.

[9] https://nest.com/

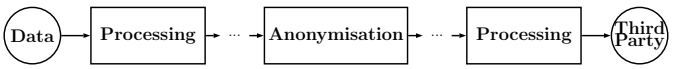

Fig. 3. Disclosure path to third party.

and archived data should not be used in a commercial endeavour);

4) detailed consumption and actuation data can be conserved for three years, but must be aggregated after this period.

IFC enforcement in the context of regulation enforcement has been discussed in previous publications [1], [18]. Here we focus on the audit aspect, and how querying an Information Flow Audit graph may be used retrospectively to verify compliance with regulation. Implementation of the enforcement and audit collection mechanism is discussed in detail in [6]. We assume that compliance data is stored in a graph database akin to Neo4J[10] and can be queried using a declarative language such as Cypher[11] or through a traversal API,[12] for example, to obtain paths between *sources* and *destinations*. Technical details are discussed in the aforementioned publications.

Recommendation 1: the sources are the customer devices, and the destinations are third party services. Verifying compliance with the first recommendation is equivalent to writing a query to find a path between a customer device and a third party service, that does not have on it an anonymisation process. If such a path exists, the data processor is in violation of the recommendations. An illustration of a path following the recommendation is presented in Fig. 3.

Recommendation 2: As discussed in [1], consent verification can be handled by a specific component. Again, if a path that does not contain a component to verify consent exists, then the data processor is potentially in violation of the recommendation. However, in some circumstances it may be difficult to reach an authoritative decision and the system may also in addition to *true* or *false* provide the response that it *cannot say*. In such situations, policy local to the decision-making component will need to determine whether to act cautiously, or prioritise potentially riskier data release.

Recommendation 3 and 4: queries can be made to verify that data used after three years are only in their aggregated form (i.e. there is no path between a commercial process and a data source older than three years without an aggregation process). Similarly, the use of data after contract termination can be verified, given the date of the termination is known.

The audit graph built by IFC&A can be exploited further in a number of scenarios. For example, as the 'Heartbleed' vulnerability [19] demonstrated, no implementation is guaranteed to be error proof, even if it is widely deployed, tested and examined. Regulators often stipulate that best practice and a state-of-the-art approach must be used, as appropriate to the sensitivity of the data. As an example, this may mean verifying that no software library version impacted by the Heartbleed

[10]http://neo4j.com/

[11]http://neo4j.com/developer/cypher-query-language/

[12]http://neo4j.com/docs/stable/tutorial-traversal-java-api.html

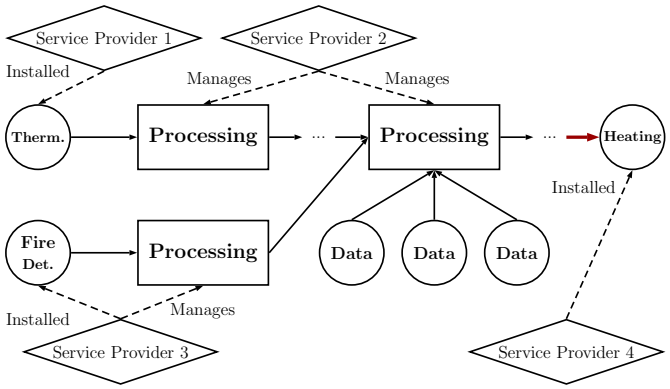

Fig. 4. Partial graph leading to a command causing a fire (red edge).

vulnerability is loaded by applications, after a reasonable period of time following publication of the vulnerability.

In the current use-case this could mean that, in addition to the existence of the anonymisation procedure, we may want to verify the algorithm and implementation version that it uses. For example, in the case of a software vulnerability being published, the compromised implementation should not be in use after a reasonable delay. Further, the graph may facilitate the identification of data that has been processed by the buggy implementation, and determination of the customers impacted by the vulnerability. The data processor may then notify them, as for example, is mandated in the US-CERT guidelines.[13]

Events in the system are represented by edges in the audit graph. For example, an actuation command is a flow (or a succession of flows) between some entity and another. If an actuation command causes physical or financial damage it may be necessary to determine who is responsible. Was the algorithm used to issue the command erroneous? Were the data captured to reach the decision inaccurate? Were errors introduced in the chain between the decision to actuate and actuation? In order to answer such questions, it is first necessary to identify the system components and person involved in the process. In the presence of a complex ecosystem, where multiple devices' manufacturers and service providers interact, this may not be a trivial task. Indeed, an actuation command should not be seen in isolation, but as the result of a potentially complex chain of events linked by causal relationships. Query of the audit graph allows this complex chain to be visualised and understood, and can help in determining where responsibility lies.

Fig. 4 presents a partial graph leading to a command that caused a fire, damaging the customer's property. While in itself it may not be sufficient to determine responsibility, it allows all parties involved to be identified and their participation in the chain of events to be explored. This has great potential in facilitating the investigation.

V. Discussion & Open Challenges

Sakka et al. [20] discuss provenance in a cloud relating to document lifecycles. The context is banking under French regulation,[14] to ensure the probative value of electronic

[13]https://www.us-cert.gov/incident-notification-guidelines

[14]Code Civil Article 1316-1.

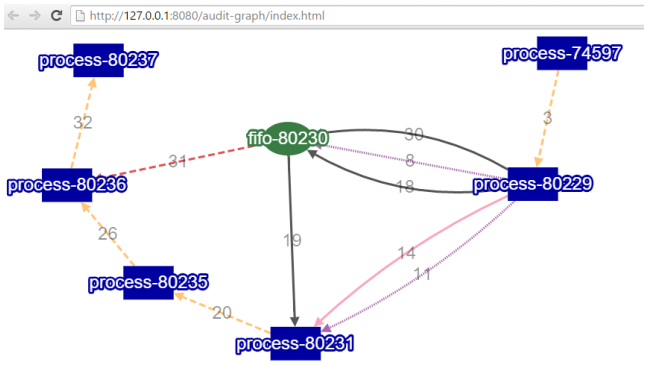

Fig. 5. Visualisation as provided by our current prototype.

documents. This requires the emitter of any document to be identified and guarantees its integrity, which is achieved through provenance in a particular closed system. Curbera et al. [21] proposed to use provenance to demonstrate compliance of businesses with regulation such as the *Sarbanes-Oxley Act* or *HIPAA*. Contrary to our approach, which captures whole-system provenance and does not require trust in involved applications, the above approaches require existing applications to be instrumented.

It is important to distinguish between observed and disclosed provenance [22]. Disclosed provenance requires applications to disclose to the audit service the data that flow through those applications. Therefore, this requires the applications to be trusted, which in the case of multi-party interaction may prove to be difficult. On the other hand, observed provenance does not require the involvement of the monitored applications and therefore trust need only be placed in the platform, making the capture (i.e. in the present work the cloud provider). However, the quality of the provenance information being captured is highly dependent on what can be observed. Earlier systems were not capable of capturing enough information to monitor all possible flows and therefore only provided a partial view of the data flow within the operating system. This may prove to be a crucial pitfall when the provenance data is to be used to demonstrate compliance. Therefore, Pohly et al. [14] proposed whole-system provenance to ensure that all data exchange between kernel objects could be observed. They leverage the Linux Security Module[15] [23] framework to provide such guarantees. A number of issues remain to be addressed: we discussed some of these issues in [6] and elaborate further on a selection of them below.

A. Data visualisation and abstraction

Fig. 5 shows how information flow audit data can be visualised in our current prototype. This figure represents different interactions (e.g. creation, change of security context, write to, read from, etc.) between entities constituting the system (e.g. processes, files, sockets, etc.). In particular it shows two IFC-constrained processes writing to a pipe, and manipulating their own security contexts. A more detailed description is available in [6]. Further information is displayed when hovering over the objects (including disclosed provenance information as reported by applications). While this may be understandable by a system engineer it may not convey any directly useful information to an end user or an auditor wishing to examine in which context a certain item of data is being used. Mechanisms need to be developed to abstract the audit graph in a manner relevant to a particular user. The generation of *super-nodes* [22] is such a mechanism. Super nodes abstract a cluster of nodes into a single node, representing a higher-level system, meaningful within the context of the policy for which the compliance is investigated. Borkin et al. [24], explored the representation of provenance as a graph or a radial plot (across multiple criteria). It is an open problem how best to represent data, so as to help users to investigate compliance through understanding the complex relationship between the entities in a system and their interactions.

[15]The Linux Security Module framework allows additional security mechanisms to be built into the Linux kernel.

B. Representing legal relationships

The IFA data model currently allows the data flows between software components to be represented. In a legal context it may also be useful to be able to represent contractual relationships as edges, thus linking legal or natural persons to each other, with system components, and with software and hardware artefacts. This may prove useful in improving transparency and helping customers navigate the complex service provision model introduced by the cloud and the Internet of Things [25], especially in scenarios where relationships can be established in a purely *ad hoc* fashion. Further, the representation of the audit data should help in understanding how interactions between entities in the system relate, or violate legal relationships. We believe that it is important to develop a way to visually understand interactions within a legal framework, and also how laws and regulations could be expressed in terms of data flow [1].

C. Provenance-based policy

As seen in the use case (§IV), some compliance requirements cannot be fully captured from simple IFC primitives (e.g. constraints applying after certain periods of time). One solution is to allow the application to specify provenance-based access control policy [26]. However, as with access control, the policy will only be enforced at a particular point in the system and it is hard to guarantee that no other path exists.

A possible solution is to force data that is destined for a third party to first flow through an element that enforces AC, using an appropriate composition of IFC enforcers/declassifiers. AC decisions could be made based on queries over the provenance engine, or by applying reasoning techniques to metadata, such as the 'baggage' used in the Pivot system [27]. The notion of 'baggage' can be seen as an extended version of taint tracking [28], where instead of simple taint, more complex metadata are attached to data items and flow with them through the system. This may improve performance, as no query over the provenance graph is required, but would necessitate prior knowledge of the metadata required to make policy decisions, which may prove difficult in a number of scenarios.

D. Addressing storage issues

A central challenge for the management of provenance-like audit data is their size. As previously discussed, storage

comes with a cost that has the potential to hinder adoption of techniques similar to those proposed in this paper. As every flow in the system may potentially be recorded, the size of the provenance data tends to grow very quickly. Approaches for pruning the data according to security policy have been proposed, based on SELinux policies [16], or our current method based on IFC policy [6]. In these two papers, data protected via policies are considered *sensitive*. Audit data are only captured for such entities, removing the "noise" generated by irrelevant, routine system operations. However, this may yet not be sufficient to make the storage cost acceptable. Further pruning techniques [13] may be used, such as deleting the audit data of an entity with no descendant when this entity is deleted, or compressing a long chain into a single node (the *super node* as described in [22]). Effective pruning would require application of research from information flow analyses and programming language 'garbage collection' techniques within this particular area. Braun et al. [22] suggest that irrelevant attributes could be deleted. Those attributes need to be identified, based on the specific context of the application. However, pruning techniques may delete information that would have proved useful for certain investigations. Therefore, such techniques need to be considered with care, especially in the context of compliance demonstration, and a balance between utility and cost needs to be established.

VI. Conclusion

In this paper we illustrated how extending Information Flow Control with provenance-like data collection can help in demonstrating compliance with regulations. We illustrated this through a realistic example provided by the French data protection agency CNIL, pointed to relevant literature on technical details, and surveyed possible approaches to extend this work. We believe Information Flow Control when augmented with an audit capability is a step towards building more transparent and trustworthy cloud services in heavily regulated sectors. We further discussed open challenges and potential candidate solutions to address them.

Acknowledgement

This work was supported by UK Engineering and Physical Sciences Research Council grant EP/K011510 CloudSafetyNet: End-to-End Application Security in the Cloud. We acknowledge the support of Microsoft through the Microsoft Cloud Computing Research Centre.

References

[1] J. Singh, J. Powles, T. Pasquier, and J. Bacon, "Data Flow Management and Compliance in Cloud Computing," *IEEE Cloud Computing Magazine, SI on Legal Clouds*, 2015.

[2] N. Kumar and R. Shyamasundar, "Realizing Purpose-Based Privacy Policies Succinctly via Information-Flow Labels," in *Big Data and Cloud Computing (BDCloud'14)*. IEEE, 2014, pp. 753–760.

[3] T. Pasquier, J. Singh, D. Eyers, and J. Bacon, "CamFlow: Managed Data-Sharing for Cloud Services," *IEEE Transactions on Cloud Computing*, 2015.

[4] M. Krohn, A. Yip, M. Brodsky, N. Cliffer, M. F. Kaashoek, E. Kohler, and R. Morris, "Information Flow Control for Standard OS Abstractions," in *Symposium on Operating Systems Principles*. ACM, 2007, pp. 321–334.

[5] C. Bier, "How usage control and provenance tracking get together-a data protection perspective," in *Security and Privacy Workshops (SPW), 2013 IEEE*. IEEE, 2013, pp. 13–17.

[6] T. Pasquier, J. Singh, , J. Bacon, and D. Eyers, "Information Flow Audit for PaaS clouds," in *International Conference on Cloud Engineering (IC2E)*. IEEE, 2016.

[7] L. Carata, S. Akoush, N. Balakrishnan, T. Bytheway, R. Sohan, M. Selter, and A. Hopper, "A primer on provenance," *Communications of the ACM*, vol. 57, no. 5, pp. 52–60, 2014.

[8] UK Information Commissioner's Office, "Data controllers and data processors: what the difference is and what the governance implications are," pp. 4–5, May 2014.

[9] R. K. Ko, M. Kirchberg, and B. S. Lee, "From System-centric to Data-centric Logging-accountability, Trust & Security in Cloud Computing," in *Defense Science Research Conference and Expo (DSR), 2011*. IEEE, 2011, pp. 1–4.

[10] Y. S. Tan, R. K. Ko, and G. Holmes, "Security and data accountability in distributed systems: A provenance survey," in *High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on*. IEEE, 2013, pp. 1571–1578.

[11] T. F. J.-M. Pasquier, J. Singh, and J. Bacon, "Clouds of Things need Information Flow Control with Hardware Roots of Trust," in *International Conference on Cloud Computing Technology and Science (CloudCom'15)*. IEEE, 2015.

[12] Y. L. Simmhan, B. Plale, and D. Gannon, "A Survey of Data Provenance in e-Science," *ACM SIGMOD Record*, vol. 34, no. 3, pp. 31–36, 2005.

[13] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. I. Seltzer, "Provenance-aware storage systems." in *USENIX Annual Technical Conference*, 2006, pp. 43–56.

[14] D. J. Pohly, S. McLaughlin, P. McDaniel, and K. Butler, "Hi-Fi: Collecting High-Fidelity whole-system provenance," in *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 2012, pp. 259–268.

[15] A. Bates, D. Tian, K. Butler, and T. Moyer, "Trustworthy Whole-System Provenance for the Linux Kernel," in *Proceedings of 24th USENIX Security Symposium on USENIX Security Symposium*, 2015.

[16] A. Bates, K. R. Butler, and T. Moyer, "Take only what you need: leveraging mandatory access control policy to reduce provenance storage costs," in *Conference on Theory and Practice of Provenance*. USENIX, 2015, pp. 7–7.

[17] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers *et al.*, "The Open Provenance Model Core Specification (v1. 1)," *Future Generation Computer Systems*, vol. 27, no. 6, pp. 743–756, 2011.

[18] T. Pasquier and J. Powles, "Expressing and Enforcing Location Requirements in the Cloud using Information Flow Control," in *IC2E International Workshop on Legal and Technical Issues in Cloud Computing (CLaw'15)*. IEEE, 2015.

[19] Z. Durumeric, J. Kasten, D. Adrian, J. A. Halderman, M. Bailey, F. Li, N. Weaver, J. Amann, J. Beekman, M. Payer *et al.*, "The matter of Heartbleed," in *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM, 2014, pp. 475–488.

[20] M. A. Sakka, B. Defude, and J. Tellez, "Document provenance in the cloud: constraints and challenges," in *Networked Services and Applications-Engineering, Control and Management*. Springer, 2010, pp. 107–117.

[21] F. Curbera, Y. Doganata, A. Martens, N. K. Mukhi, and A. Slominski, "Business provenance–a technology to increase traceability of end-to-end operations," in *On the Move to Meaningful Internet Systems: OTM 2008*. Springer, 2008, pp. 100–119.

[22] U. Braun, S. Garfinkel, D. A. Holland, K.-K. Muniswamy-Reddy, and M. I. Seltzer, "Issues in automatic provenance collection," in *Provenance and annotation of data*. Springer, 2006, pp. 171–183.

[23] C. Wright, C. Cowan, S. Smalley, J. Morris, and G. Kroah-Hartman, "Linux Security Modules: General security support for the Linux kernel," in *Foundations of Intrusion Tolerant Systems*. IEEE, 2003, pp. 213–213.

[24] M. Borkin, C. S. Yeh, M. Boyd, P. Macko, K. Z. Gajos, M. Seltzer, H. Pfister *et al.*, "Evaluation of filesystem provenance visualization tools," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 19, no. 12, pp. 2476–2485, 2013.

[25] J. Singh, T. Pasquier, J. Bacon, H. Ko, and D. Eyers, "Twenty security considerations for cloud-supported Internet of Things," *IEEE Internet of Things Journal*, 2015.

[26] A. Bates, B. Mood, M. Valafar, and K. Butler, "Towards secure provenance-based access control in cloud environments," in *Proceedings of the third ACM conference on Data and application security and privacy*. ACM, 2013, pp. 277–284.

[27] J. Mace, R. Roelke, and R. Fonseca, "Pivot tracing: Dynamic causal monitoring for distributed systems," in *25th Symposium on Operating Systems Principles (SOSP '15)*. ACM, 2015.

[28] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones," in *Conference on Operating systems design and implementation (OSDI'10)*. USENIX, 2010, pp. 1–6.